Guia Linux

AWK

awk [padrão ação textos]

Descrição

- A linguagem de programação AWK permite a manipulação de textos a partir de uma sequência de padrões. É uma linguagem interpretada (não há necessidade de compilação).
- O nome AWK vem dos sobrenomes dos três criadores da linguagem: Alfred V. **A**ho, Peter J. **W**einberger e Brian W. **K**ernighan.

Algumas opções do comando

- **-f** : especifica o nome do arquivo com os comandos a serem executados.
- **-F** : define quem é o separador de campos (o padrão é o espaço).
- **-h** ou **−−help** : informa as opções do utilitário.
- **-V** ou **−−version** : informa a versão do utilitário.

Exemplos

- Suponha que queremos obter a lista dos usuários do sistema e o **shell** usado por cada um dos usuários. Estas duas informações fazem parte do arquivo */etc/passwd* onde existe uma entrada para cada usuário do sistema com as seguintes informações: nome, senha, UID, GID, informações do usuário, diretório home e o *shell* padrão. Por exemplo, a linha abaixo mostra a entrada para o usuário *aluno* que usa o *shell bash* como padrão.

```
aluno:x:1001:1001::/home/aluno:/bin/bash
```

Note que os campos são separados por dois pontos (":"). Podemos então, digitar o seguinte comando para obter os nomes e os *shells* dos usuários:

```
awk -F: '{print $1 " => " $7}' /etc/passwd
```

A opção -F define o caractere ":" como separador dos campos. O comando *print* define a exibição de três argumentos: o primeiro campo de cada linha ($1), a string " => " e o sétimo campo de cada linha ($7). Portanto, a saída do comando acima para a entrada do usuário *aluno* é

```
aluno => /bin/bash
```

- Também podemos executar o comando abaixo para obter o mesmo resultado do exemplo anterior.

```
awk -F: -f lista_usu /etc/passwd
```

onde *lista_usu* é um arquivo texto com o seguinte conteúdo

```
{print $1 " => " $7}
```

- Outra forma é criar um arquivo e torná-lo executável com o comando **chmod**. Por exemplo, suponha que o arquivo *teste* possui o conteúdo mostrado abaixo. Note que agora o separador de campos é definido na variável FS (*Field Separator*) e que uma frase é exibida antes de processar o arquivo de entrada (bloco BEGIN) e outra frase é exibida no final da execução (bloco END). Apenas os comandos do bloco central são executados de forma repetitiva (ler uma linha do arquivo de entrada e executa o bloco central). Portanto, neste exemplo, o número de vezes que o comando *print* do bloco central é executado depende do número de linhas do arquivo */etc/passwd* usado como entrada.

```
#!/usr/bin/awk -f
BEGIN { FS=":"; print "*** inicio ***\n" }
  {
    print $1 " => " $7
  }
END { print "*** fim ***" }
```

Antes de executá-lo, devemos alterar a **permissão** do arquivo comchmod u+x testeEm seguida, basta digitar

```
./teste /etc/passwd
```

O resultado será o mesmo dos dois primeiros exemplos, exceto que aparecerão duas frases: uma no início e outra no fim da execução.

Mais Exemplos

- Ler o arquivo */etc/passwd* passado como parâmetro

```
BEGIN { FS=":" }
{
  for (i = 1; i < ARGC; i++)
```

```
    print $1
}
END { print "*** FIM"}
```

- O exemplo lista as variáveis de ambiente

```
BEGIN { print "*** INICIO" }
{
  OFS="="
 for(i in ENVIRON)
    print i, ENVIRON[i];
}
END { print "*** FIM"}
```

- Este exemplo mostra o uso de parâmetros e variáveis.Por exemplo, o comando

```
awk -v A=1 -f teste B=300 /dev/null
```

recebe a variável A (com opção -v), o parâmetro B=300 (uma string) e um parâmetro /dev/null (para indicar o fim do comando).

```
BEGIN {
C = 2000
printf "A=%d, B=%d, C=%d\n", A, B, C
printf "ARGC = %d\n", ARGC
for (i = 0; i < ARGC; i++)
printf "\tARGV[%d] = %s\n", i, ARGV[i]
}
END { printf "\n*** A=%d, B=%d, C=%d\n", A, B, C }
```

Abaixo o resultado da execução do *awk*.

```
A=1, B=0, C=2000
ARGC = 3
ARGV[0] = awk
ARGV[1] = B=300
ARGV[2] = /dev/null

*** A=1, B=300, C=2000
```

O uso da opção -v antes do parâmetro B=300 faz com que B seja definido como variável antes da execução. Neste caso, o programa só teria /dev/null como parâmetro.

- O exemplo abaixo exibe o resultado da soma de dois valores recebidos na linha de comandos. Da mesma forma da função *main( )*, as variáveis ARGC e ARGV contém, respectivamente, o número de parâmetors e os argumentos passados na linha de comandos (o parâmetro ARGV[0] possui "awk"). Observe que todos os comandos estão dentro do bloco BEGIN (isto significa que os comandos serão executados uma única vez).

```
#!/usr/bin/awk -f
BEGIN {
  if (ARGC != 3)
    print "Forneca 2 numeros\n"
  else
    {
      soma=ARGV[1]+ARGV[2]
      print "Soma = " soma
    }
}
```

- Abaixo é apresentado um exemplo que calcula o fatorial de um número passado na linha de comandos.

```
#!/usr/bin/awk -f
BEGIN {
  fat = 1
  for (i = 1; i <= ARGV[1]; i++)
    fat = fat * i
  print "Fatorial de " ARGV[1] " = " fat
}
```

- O exemplo abaixo calcula a raiz quadrada de um número. Note que o programa verifica se o parâmetro passado pelo usuário é realmente um número.

```
#!/usr/bin/awk -f
BEGIN {
  if (ARGC != 2 )
    print "Forneca um numero"
  else
    {
      if (ARGV[1] ~ /^[0-9]+$/)
        {
          raiz=sqrt(ARGV[1])
          print "Raiz quadrada de", ARGV[1], "=", raiz
        }
```

```
        else
          print "Valor nao numerico"
      }
}
```

- Um exemplo de uso de *arrays* é mostrado a seguir. Note que o índice do *array* não precisa ser um valor numérico.

```
#!/usr/bin/awk -f
BEGIN {
  vetor[0] = 1
  vetor[1] = 2
  vetor[2] = 3
  dicionario["teste"] = "socorro"
  dicionario["AOU"] = "Linux"
  for (i = 0; i < 3; i++)
    print "vetor[" i "] = " vetor[i]
  print "teste = " dicionario["teste"]
  print "AOU = " dicionario["AOU"]
}
```

- O exemplo abaixo mostra outra forma de ler o arquivo */etc/passwd* e exibir o nome do usuário (primeiro campo = $1) e o *shell* usado pelo usuário (sétimo campo = $7). Os nomes dos usuários são usados como índices do vetor e cada posição do vetor armazena o *shell* do usuário. A variável NF contém o número de campos da linha que, neste caso, é igual a 7.

```
#!/usr/bin/awk -f
BEGIN { FS=":" }
{
  usuario[$1]=$NF
}
END {
  for (i in usuario)
    print i,"=>", usuario[i]
}
```

- O programa *teste* mostrado abaixo faz a multiplicação de duas matrizes armazenadas em arquivos distintos (os nomes dessas matrizes são fornecidos na linha de comandos). Observe que o comando *getline*, usado para ler uma linha de arquivo, retorna 1 quando é executado com sucesso e zero quando nenhuma linha é lida. Neste caso, $0 corresponde à linha toda lida, $1 corresponde ao primeiro campo da linha, $2 ao segundo campo da linha, e assim por diante.

```
#!/usr/bin/awk -f
BEGIN{
  # Ler primeira matriz
  valor = getline < ARGV[1]
  lin_A = 0
  while (valor == 1)
   {
     lin_A++
     for (i = 1; i <= NF; i++)
       A[lin_A][i] = $i
     valor = getline < ARGV[1]
   }
  col_A = NF
  # Ler segunda matriz
  valor = getline < ARGV[2]
  lin_B = 0
  while (valor == 1)
   {
     lin_B++
     for (i = 1; i <= NF; i++)
       B[lin_B][i] = $i
     valor = getline < ARGV[2]
   }
  col_B = NF
  if (col_A != lin_B)
    printf “Numero de colunas de A dever ser igual a numero de linhas de B\n”
  else
   {
     for (i = 1; i <= lin_A; i++)
       {
         for (j = 1; j <= col_B; j++)
           {
             C[i][j] = 0;
             for (k = 1; k <= col_A; k++)
               {
                 C[i][j] += A[i][k] * B[k][j]
               }
             printf C[i][j] ” ”
           }
         printf “\n”
       }
   }
```

```
}
```

Suponha que o arquivos A e B tenham os conteúdos mostrados abaixo.

Arquivo A

```
1 3 1
2 3 2
```

Arquivo B

```
2 1
3 1
1 2
```

Ao executar o programa com

```
./teste A B
```

o AWK fornece a seguinte resposta:

```
12 6
15 9
```

Variáveis internas

O AWK possui variáveis internas que podem ser usadas pelo programador. A tabela abaixo mostra algumas dessas variáveis.

Variável	Descrição
ARGC	Número de argumentos da linha de comandos
ARGV	Vetor com os argumentos da linha de comandos
ENVIRON	Vetor com as variáveis de ambiente
ERRNO	Armazena mensagem de erro
FILENAME	Nome do arquivo de entrada sendo processado no bloco central
FNR	Número de linhas já processadas de um determinado arquivo
FS	Separador dos campos da linha de entrada (o espaço é o padrão)

NF	Número de campos na linha sendo processada
NR	Número de linhas já processadas considerando todos os arquivos de entrada

Observações

- A linguagem AWK é muito rica pois permite, por exemplo, o uso de
  - comandos de decisão (if, else);
  - laços (for, while, do while);
  - expresões aritméticas (+, -, *, /, %, ++, −−, =, +=, -=, >, >=, <, <=, !=, etc);
  - funções (exp, log, sqrt, split, substr, getline, etc);
  - variáveis, constantes e vetores.

  Para obter mais informações, consulte o **manual on-line** (digite “man awk”).

- O *mawk* é o interpretador para a linguagem AWK. Veja o site oficial do **mawk** ou consulte o manual on-line.
- O *gawk* é é a implementação do Projeto GNU para a linguagem AWK. Veja o site oficial do projeto **gawk** ou consulte o manual on-line.

Sumário | **Topo**